x00111602

IT TALLAGHT  Conor Griffin

CA2: Detect patients that have asd

report by: conor griffin x00111602

Contents

[1. Section 1 2](#_Toc510449449)

[1. Analyse the dataset 2](#_Toc510449450)

[2. Prepare a number of views (formats) of the dataset 3](#_Toc510449451)

[Original Dataset 3](#_Toc510449452)

[Normalised View 3](#_Toc510449453)

[Standardised View 3](#_Toc510449454)

[Missing View 4](#_Toc510449455)

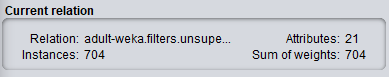
[3. Attribute Selection (Feature selection) 5](#_Toc510449456)

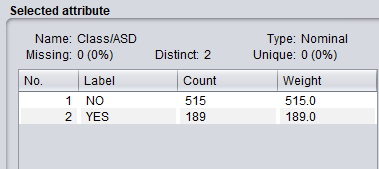
[4. Use kNearestNeighbor (IBK) Classifier on the dataset 8](#_Toc510449457)

# Section 1

## Analyse the dataset

From looking at the dataset in Weka explorer there are a few things to take notice of. Looking at the current relation section in weka explorer it tells us that there are 704 rows of data and 21 columns/attributes in the dataset. For the attributes question 1-10 with a scale value of nominal type, each attribute has no missing values and no unique values, also each attribute has 2 distinct values. Another thing to take note of is that there are only 2 numeric attributes in the dataset, this can be a negative aspect in a dataset for reaching conclusions as it is good to have numeric attributes because they give a machine learning algorithm constant and final values to use. For each of the numeric attributes they have almost no missing values, age has just 2 missing values. Also the result numeric attribute has a minimum 0, maximum 10, Mean 4.875, Standard deviation 2501 this gives an idea of the spread of data. The attribute relation has a high amount of missing values 95 (13%) this will be interesting to see how algorithms interpret this amount of missing data and will this ratio of missing data mean I will have to remove or impute the values. The class attribute determines if an adult is autistic, “Yes”, or not, “No”, there is a higher count of patients in the No category 515 and 189 Yes. I would say that this is an accurate dataset in the sense you would expect to see a higher count in the no category based off the 10 questions.

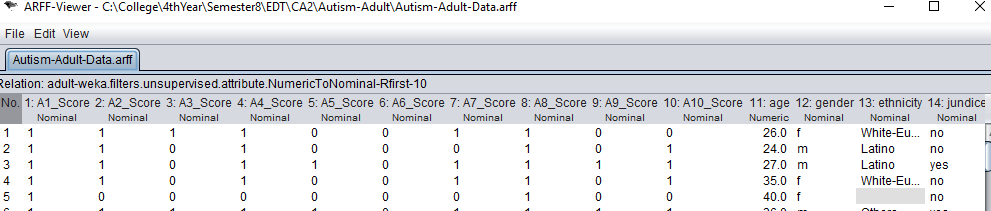




## Prepare a number of views (formats) of the dataset

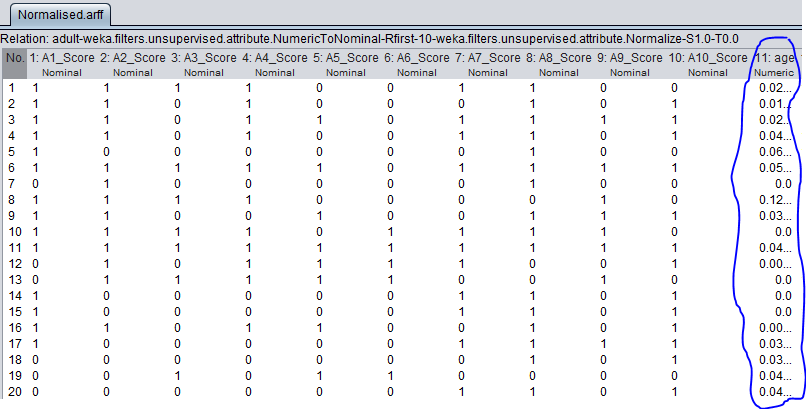
### Original Dataset

I loaded the dataset into Weka Arff viewer.



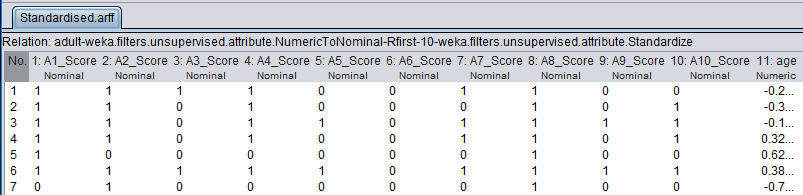
### Normalised View

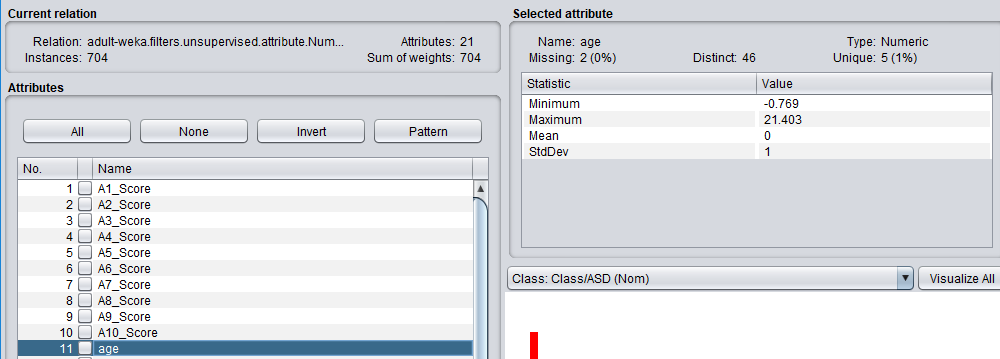
To normalise the view I first created a duplicate of the original dataset, then loaded it into weka and began the process in weka explorer. Data normalisation is the process of rescaling one or more numeric attributes to the range of 0 to 1. This is a good technique to use when you do not know the distribution of your data or when it is not a bell curved distribution. In weka explorer I chose the normalize filter and applied it to the autism dataset, renamed to normalised. From the final view you can see that the age numeric attribute has been normalised from ranges 0 to 1.



### Standardised View

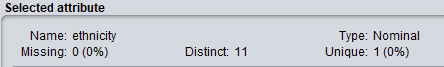
To standardise the view I began by creating a copy of original dataset called standardised.arff and loaded it into weka explorer. Data standardisation is the process of rescaling one or more attributes so that they have a mean value of 0 and a standard deviation of 1. I standardised each of the attributes in weka by choosing the standardize filter and applied it accordingly. From the view you can see that the numeric attributes have been standardised, also in explorer you can see for example the age attribute has a mean of 0 and standard deviation 1.



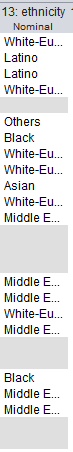


### Missing View

To impute the average into attributes with missing data I first loaded the dataset into weka and created a missing.arrf file. I then loaded it into weka explorer and used the ReplaceMissingValues filter. The results replaced all missing values with the average value. For example you can see below the attribute ethnicity’s values were imputed with the average ethnicity.



Normal view Missing View

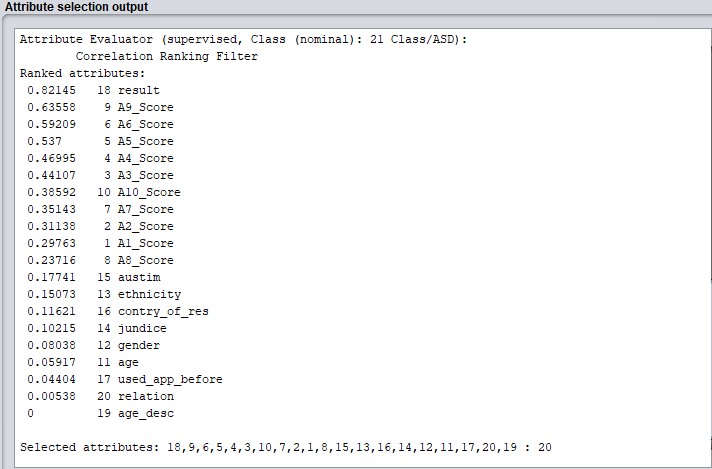
 

## Attribute Selection (Feature selection)

For attribute selection I divided the process into two parts: Attribute evaluator, this is used to evaluate the attribute in the context of the class variable. The Search Method, the technique to try or navigate combinations of attributes in the dataset in order to get on a short list of chosen features.

The first attribute selection method I used to determine which attributes to choose is the correlation attribute evaluation technique. I used this method with a Ranker search method which evaluates each attribute and lists the results in rank order. I configured both the attribute evaluator and search method to work with each other in Weka. I ran the algorithm on the dataset and go the following results.

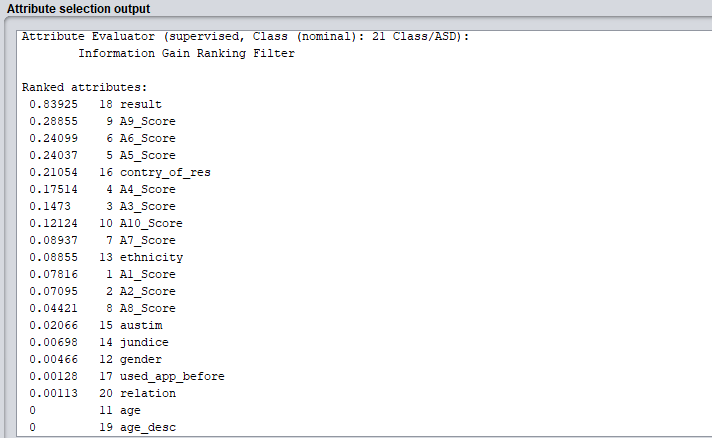




The technique to use here is to take a look at the correlation in attributes. Correlation is calculated for each variable predictor variable and the process to follow is to select only those attributes that have a moderate to high positive or negative correlation (close to -1 or 1) and drop those attributes with a low correlation (values close to 0). The use of the ranker search method also displays the attributes in a ranked order. Looking at the output you can see that the predictor attribute result has a really high correlation (0.82145) with the output class, also A9\_Score, A6\_score, A5\_Score, A4\_Score and A3\_Score have a high correlation with the class/ASD variable. If we set the cut-off point at 0.4 for relevant attributes then the remaining 14 attributes could possibly be removed, but this seems like a lot of data to remove so I would either set the cut-off lower or use other techniques to make a final decision on what attributes to remove.

The second method I used for attribute selection is the Information gain technique, which is used to calculate the info gain for each attribute for the output variable ASD. Entry values from 0 (no information) to 1 (maximum information). Those attributes that contribute more information will have a higher information gain value and can be selected whereas those that do not add much information will have a lower score and can be removed. In Weka I use the InfoGainAttributeEval in attribute evaluator and again use the Ranker search method. The following output displays the results.

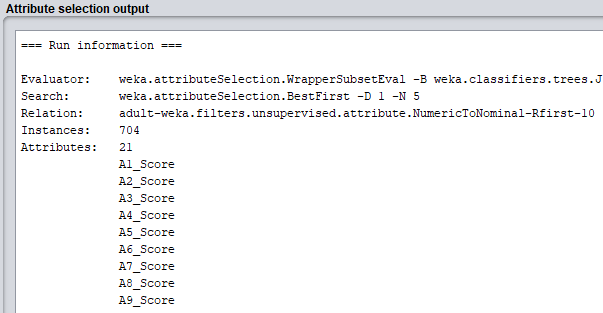


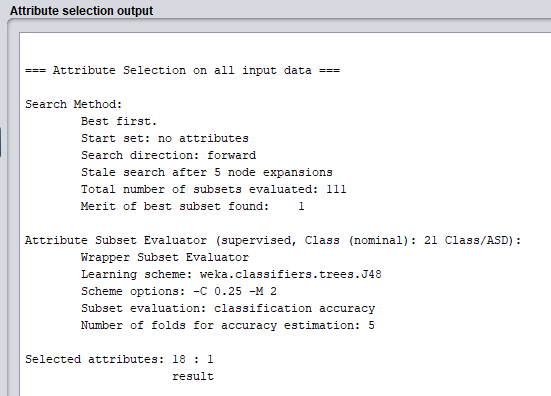


From the output you can see that the top ranked attribute is Result with a high information gain of 0.83925, most of the other attributes have a low info gain, but if we set the cut-off point at 0.2 then we could keep other attributes like A9, A6, A5\_Score and contry\_of\_res and the rest could be removed from the dataset.

The third selection method I used to determine what attributes to use in the modelling stage is the learner based feature selection. This is a powerful learning algorithm and is used to gather subsets of attributes, the subset with the best results and performance is taken as the selected subset. The feature selection method used is the WrapperSubsetEval technique and uses a BestFirst search method as it uses less compute time. I modified the configuration to instead of ZeroR, use J48 in trees, this gives a better preferred subset in the output.

The run information for this algorithm tells us that it ran the subset method with a best first search and used 704 instances over 21 attributes, it found 111 subsets and the merit of the best subset found was 1. The final selected attributes was attribute 18 result.

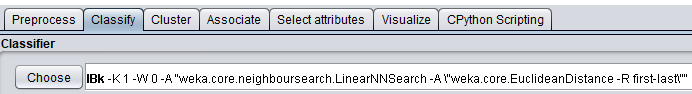




Looking back over the three techniques, you can see that a few of the attributes overlapped into each of the methods used. The results attribute would be the strongest and best attribute to keep in the dataset based on the three techniques used as it was number 1 in the output of each algorithm. Most of the other attributes didn’t perform well in each of the feature selection methods but this doesn’t necessarily mean you can get rid of all them, I would say it best to keep attributes that performed well in the first two techniques like A9\_Score, A6\_score, A5\_Score, A4\_Score and A3\_Score as well as contry\_of\_res attributes. I would add each of these in the modelling stage.

## Use kNearestNeighbor (IBK) Classifier on the dataset

Nearest neighbour: to classify a new instance, search training set for one that’s most like it. Searches the training set for the one that most like the new instance. We have our class variable with a yes or no answer, we now have the unknown instance, and the nearest neighbour method produces the new instance based on previous data most similar to it. It uses a similarity function by measuring the Euclidean distance, Manhattan distance, normalize the attributes to be between 0 and 1 and this gives us the kNearest Neighbor. KNearest protects us from noisy data, we choose the k nearest neighbors and it chooses the majority class among the neighbours.



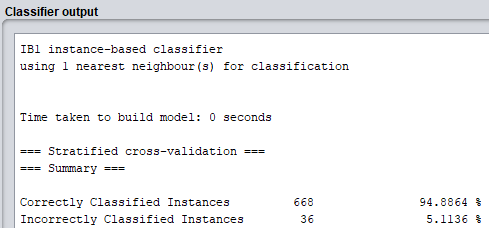
First, I found the baseline accuracy for the dataset, I ran the rules->ZeroR algorithm on the dataset, and this gave an output of 73.15%. I was able to base my accuracy of kNearest results off of the ZeroR result of 73%.

ZeroR

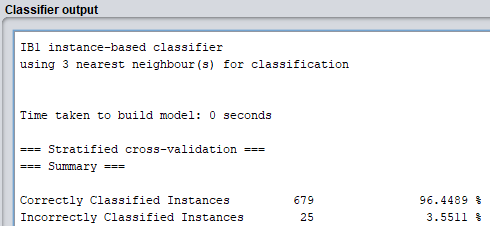


If any of the values for k have a greater accuracy than baseline performance then the more optimal that solution is. For k=1 the number of correctly classified instances was 668, incorrect classified instances was 36, with a 94% accuracy, k=3 the number of correctly classified instances was 679, incorrect instances was 25 with a 96% accuracy, k=5 the number of correctly classified instances was 675, incorrect instances was 29, with an accuracy of 95%, k=7 the number of correctly classified instances was 685, incorrect instances was 19, with an accuracy of 97% and k=15 the number of correctly classified instances 690, incorrect instances was 14, with an accuracy of 98%.

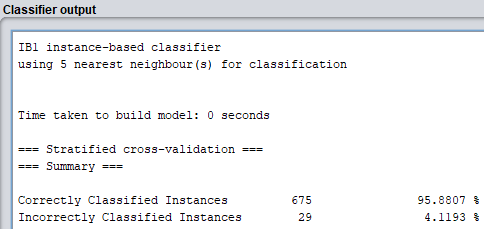
K=1



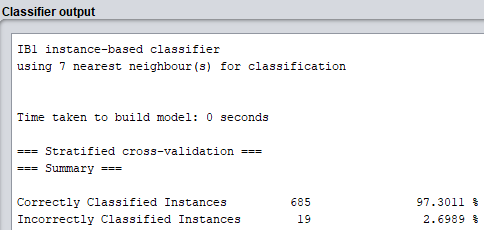
K=3



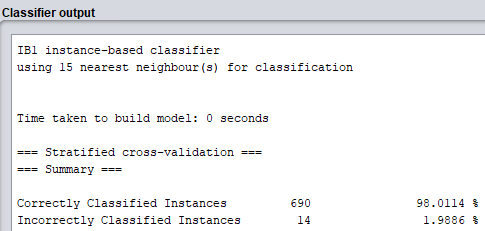
K=5



K=7



K=15



The optimal value of k I would say from looking at the results is 15 as it gave the most accurate % classified instances and more instances were produced (690). It seems that from looking at the results the greater the value for K then the better and more accurate the output but this is not the case, I also tested for greater values of k such as 20 and 50, the accuracy of these models were less than the model of k=15. Using 20 nearest neighbours for classification produced 689 correct instances and a 97% accuracy, using 50, produced 686 correct instances and 97% accuracy, both models less accurate than using 15 nearest neighbours for classification.

## Two Machine Learning Algorithms for Evaluation